Scramikub

Final Report

Author: Ismael Valenzuela García
Director: Enric Mayol Sarroca
Facultat d'Informatica de Barcelona (FIB)
Universitat Politecnica de Catalunya (UPC) - BarcelonaTech
April 2018

Resum

Scramikub és un joc de taula que és el resultat de la unió de dos jocs ja existents, l'Scrabble i el Rummikub. En aquest projecte, millorem l'aplicació mòbil que adapta aquest joc. Les millores més importants a realitzar i en les que ens centrarem més són: fer les partides parametritzables, crear un intel·ligència artificial per a jugar en contra i fer canvis en interaccions amb els usuaris. Per desenvolupar-lo, utilitzem la nova versió del framework que el que es va utilitzar per a la primera versió de l'app(Ionic).

Resumen

Scramikub es un juego de mesa que es el resultado de la unión de dos juegos ja existentes, Scrabble y Rummikub. En este proyecto, mejoramos la aplicación de móvil que adapta este juego. Las mejoras más importantes a realizar y en las que nos centraremos más son: hacer las partidas parametrizables, crear una inteligencia artificial para jugar en contra y hacer cambios en interacciones con los usuarios. Para desarrollarlo: utilizamos la nueva versión del framework que se utilizó para la primera versión de la app (Ionic).

Abstract

Scramikub is a board game which it’s the result of the union of two existent games, Scrabble and Rummikub. In this project, we upgrade the mobile application that adapts this game. The most important upgrades and the ones that we are going to focus are: to make parameterizable games, to create an artificial intelligence in order to play against and to make changes in some interactions with the users. To develop it, we use the new version of the framework used for the first version of the app.

Contents

1. Context

1.1. Introduction

Scramikub is a board game that is born of the combination of two other games: Scrabble and Rummikub. It was design by Alex González Rodríguez in collaboration with Enric Mayol Sarroca. In this project, we are going to adapt Scramikub into a mobile application, for both Android and iOS devices.

1.2. Scramikub, The Board Game

As we noted before, Scramikub is the mixture of two famous games: Scrabble and Rummikub. It keeps the basic rules of Rummikub (you need to create a series of numbers with the numbered tokens given to you). But it adds another difficulty layer by making you to place them in a board similar to Scrabble.

Scramikub consist of:

A board of 15 per 15 holes in it, where tokens are supposed to go.

1-minute, 2-minute and 3-minute hourglasses.

30 blue chips numbered from 1 to 15.

30 red chips numbered from 1 to 15.

30 dark green chips numbered from 1 to 15.

30 brown chips numbered from 1 to 15.

30 purple chips numbered from 1 to 15.

30 light green chips numbered from 1 to 15.

6 black joker chips.

Scramikub game mode and rules:

- A Scramikub game can house from 2 to 6 players.

- Scramikub player’s target is to get rid of the chips given to him at the beginning of the game or in-game.

- In order to do so, he has to play his chips making series.

- There is only two ways of making series:

- Series composed of consecutive numbers in the same color (chip numbers are supposed continuous; 15 is followed by 1).

- Series composed of the same numbers but in different colors.

- The sum of the numbers of the sequence has to be above 15.

- To make a sequence the player have to use three or more chips.

- At the beginning of the game, 15 random chips are given to every player.

- Then 1 more chip is given to each player, the one with the higher chip will be the first to play.

- The next player will be the one on the right.

- In his turn, the player has a minute to make a sequence and place it on the board. Once he has done the move, he is able to reorganize the chips he wants of the board respecting the rules of making series stated before.

- If the player tried reorganize any chip and wasn’t successful before his minute ends, he has to keep those chips.

- If a player isn’t able to make a move in his turn, a chip will be given to him. If he is now in position of making a move, he can proceed. Otherwise, the turn goes to the next.

- The ends of the board are continuous. Hole A15 is next A14, A1, B15 and O15, for example.

- Wildcard chips can replace every colored token but it only can take one value at a time.

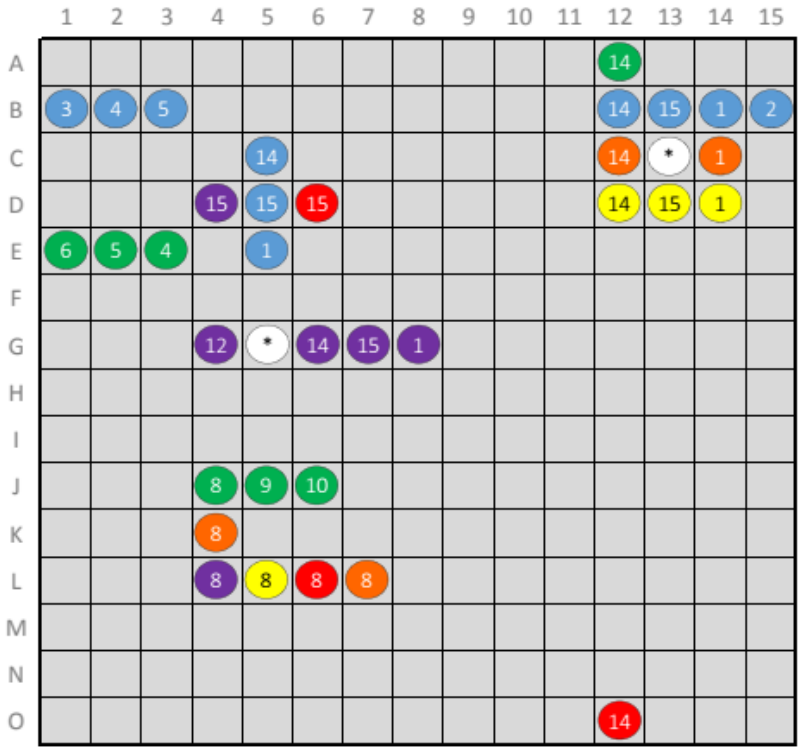


Figure1. A valid board.

1.3. Stakeholders of the application

Players: They are final users of the application and the main stakeholders. It can be a wide audience made of different kinds of users since the recommended age for playing the game is +8 years old and the fact that every day there are more children that have access to mobile devices.

Google Play and App Store: These are the markets where the application will be distributed. Our application has to satisfy the requirements given by these platforms for the purpose of been distributed by them.

Scramikub's creators: They designed the game and its mechanics and rules. They are significantly important in the process of adapting their game into mobile devices.

Developer: The person responsible of adapting Scramikub game into a mobile application.

2. State-of-art

The main goals of this section are showing in which state is the mobile games industry, displaying how possible competitors have developed other board game adaptations and also, inferring which are the current tendencies in mobile game development.

2.1. Mobile Games Industry

Nowadays, mobile industry is in a growth period. In particular, last year mobile videogame sales surpassed computer games sales or console games. This same year, 2016, it was estimated that mobile games obtain 37% of the videogame market share [1] (and it is expected to rise to 42% this year[2]) , leaving behind the computer sector with a 32% and the console sector with 31%. Mobile market imposed on the rest of the platforms due to, basically, mobile devices are more extended than their competitors and also thanks to the fact that mostly of the mobile videogames are free to play or with low prices that make them more affordable than the rest of the platforms.

In order to make an idea of the volume of the market that the mobile gaming handles, it is calculated that on the beginnings of 2017, Niantic, with the popular PokemonGo which is free, had an income of 117€ per minute [3], thanks to in-game micro-transactions. Seeing that data, it is easy to understand why everyday giant enterprises are investing in the sector.

2.2. Similar Games

Being Scramikub the fusion of Scrabble and Rummikub (see paragraph 1.2) is sensible to think that the derived applications of these games will be the direct competitors of our project. The next four exposed games are main application to take into account:

Scrabble

Scrabble is the official application of the Scrabble board game. It is developed by the well-known company, Electronics Arts. It is available for iOS and Android. The game interface tries to remind us the board of the original game, making use of different tonalities of green. Currently, it has a free version for iOS and Android and a premium version, which costs 6.95€, for iOS. Free version relies on ads between turns and in-game transactions and premium relays on purchases. It has various game modes such as playing with friends, playing against someone random, playing against the CPU and playing in the same device. It has more than 5,000,000 downloads on android.

Apalabrados

Apalabrados is elaborated by Etermax and it is the most extended mobile game based in scrabble, at least for the Spanish-speaking community. The game interface is very similar to the Scrabble application, but the color palette is more grateful to the sight. The application is available and free for both for iOS and Android and use similar techniques as Scrabble to monetize downloads. Also, this adaptation only has 2 game modes: play with a friend and play against someone random. It has more than 10,000,000 downloads on android.

Rummikub

Similar to Scrabble, Rummikub also has its official application Made by Kinkajoo, it is a paid application available for iOS (3’45€) and Android (2’99€). This application also suggests the original board game using same tokens and a similar color palette. It has more than 100,000 downloads on android.

RummyPlus

Keeping with the similarities of the applications between Rummikub and Scrabble, Rummikub also has a non official application that has more users than the original. RummyPlus has more downloads than the original Rummikub due to the fact that is free. It is developed by Peak Games and is available for both operating systems (free for both too). The main difference between this application and other free application is that it has not ads. Also, this application has an interface very similar to the Rummikub app. This app offers only two game modes: play with friends and play against someone random. It has more than a 1,000,000 downloads on android.

As we can see all of them have a great number of users. If we can manage to attract some of them to our application, it could be a good hit. Also, we want to highlight that all of them use muted color palettes and a slow paced play-style to make the experience more relaxing.

3. Scope

3.1. Project Goals

As we had said before, this project consists of making an adaptation for mobile devices of a board game. In particular, we are going to upgrade the older version of Scramikub. The improvements we propose are:

- **The capability of creating and playing parameterizable games**. We think that is interesting to give to the players the possibility to create custom games. This will make our application richer and more diverse and it will make players to find what type of games they like the most. Few of the thought parameters are number of players, turn time, use of jokers.

- **Developing an Artificial Intelligence** (AI), in order to make players practice and play against the machine. And also, it is possible that we get to a state that this becomes a challenge to the players.

- **Making a revision of the old interfaces and user interactions**. We are going to enrich old interactions in order to make them more comfortable and satisfying for the players.

- **Changing the game chat.** We will make chats easier to access so the players can have the option of chatting without loading the game.

The technical competences of this project that we aim to accomplish are:

- **Identifying, evaluating and managing potential risks associated to the construction of software.** As every project needs to deal with risks we want to get knowledge and experience about how to manage it.

- **Developing, maintaining, and evaluating services and distributed applications with network support**. Since the project it is to develop a mobile client-server application which use service architecture to communicate.

- **Defining and managing the requirements of a software system**. Before beginning to implementing, it is necessary to define which requirements our project needs to meet. Because there is a previous version of scramikub part of the requirements ara already. Medium achievement level.

- **Designing appropriate solutions in one or more domains of the application, using software engineering methods that integrate ethics, social, legal and economical aspects.** In our project we need to design a multi-platform application that could be published so it needs to be integrated with all this aspects.

3.2 Possible Obstacles

The main obstacles of this project are:

Time Restriction: TFG is time bounded in a short period of intensive work. Non work related distractions and illness could be major problems in order to finish it.

Ignorance of technology: We will work with technologies that we are not familiar with and getting used to it could cause great losses of time.

4. Methodology and validation

Given the relative short period of time available for finishing the project. We have decided to use an agile methodology for our project.

Firstly, we thought about using Scrum. Scrum is an agile methodology and as such, it was created to get as quickly as possible good results where requirements are changing and the flexibility is key. Scrum organizes tasks in sprints. A sprint is an iteration on the development of the product [8]. Each iteration is meant to deliver a completely finished increment of the final product. At the end of every sprint will produce an independent unit of software the will work as a black box. Working with this in mind, it is mandatory to completely test every developed user story in other to be able to affirm that is finished software.

Later, we discarded Scrum because we realized that is heavily focused on team development and this is a single-developer project. So we finally decided to adapt a variation of scrum created for teams of one people called Scrum of One. Scrum of one presents good practices and brings us a framework that match perfectly with our philosophy.

In our project, every sprint begins with a sprint plan. In this sprint plan, we plan about we are going to do in this sprint having present what we have done in the past sprint. This is specially important in sprints where the tasks are related (for example sprint 2 and 3). Then, at the beginning of each day of the sprint we plan about what we are going to today (do we need to end something from yesterday?, what is the next user story we are going to focus?). Every Friday we make a retrospective in order to see how the sprint is evolving and to act accordingly. Finally, at the last day of the sprint we evaluate how the sprint have gone.

5. Temporal Planning

5.1. Calendar

The project duration was expected to be of 4 months, beginning in the middle of September when GEP begins and ending in early February with the final release. But due to two unexpected workload peaks from the developer's another job and some complications during the development we had to extend the duration for another 3 months.

5.2. Task Description

5.2.1. Initial Inception

This is the task that is labored during GEP subject. It tries to establish the basis of the project by investigating about its context, defining its scope and time duration and organizing the pertinent tasks and resources. It went from 18/09/17 to 23/10/17 approximately.

5.2.2. Building and Prioritizing the Product Backlog (Sprint 0)

Following the Scrum methodology, it is needed to form a Product Backlog which it will collect the user stories that will tell us what it must be done and what are truly the priorities. User stories, that can consist in requirements, features or bug fixes, defined here are able to be modified or disaggregated into smaller yet more detailed user stories. User stories will be prioritized in a way that the ones with a return of inversion (ROI) higher will have a higher priority too. Another criterion that will help us is the MoSCoW criteria (Must-Should-Could-Would). User stories are located in the 8.1. paragraph. This phase went from 24/10/17 to 31/10/17.

Once it is surpassed the initial phase of the project, each of the next tasks will be treated in different Sprints.

5.2.3. Creating Parameterizable Games (Sprint 1)

The goal of the first Sprint will be the creation of parameterizable games (user story #1 see 8.1) in such a way the users can customize the games in their favorite way. There will be various parameters that the user can choose like the number players, dimensions of the board, number of jokers, number of colors between others. This parameters will be explained later in the paragraph 9.4. We have decided that this task is done in the first sprint given that some parameters can influence in the development of the AI.

In this sprint, our work will be related to do an analysis of every aspect of the game that could be parameterizable, to design the system that will allow this parameters, to implement this system and, to test and deliver the product.

This sprint became bigger by 3 more weeks because of one of the peaks of work of the developer. The sprint duration was from 01/11/17 to 17/12/17 approximately.

5.2.4. Developing an AI player (Sprint 2 y 3)

The objective of the second and third sprint was the development of an AI (user story #2) that will give to the player the possibility of practicing against the machine. We calculated that this task will be larger than the rest due to the fact that it will be the most difficult part of the project and a proper investigation in the field is needed. We decided to divide two sprints: one for the creation of a prototype of the AI and the other in order to improve it.

The activities to do in the first sprint of the 2 are: learning about artificial intelligence developing and deepening in Scramikub's mechanics, analyzing the requirements of the AI, designing the AI and, implementing, testing and delivering it. In the second sprint we will refine this AI to get a better overall performance.

At the time we arrived to the second sprint, we realized that we had to redo the evaluation process of the player's moves in order to bring more information to our upcoming AI so we decided to dedicate the second sprint to this task (user story #2.1) . Because of the second work peak of the developer, this sprint was almost split up in two going from 18/12/17 to 26/02/18.

In the third sprint, we compressed the original sprint 2 and 3 (user story #2.2) because of the redoing of the evaluation process. This sprint went from 27/02/18 – 19/03/18.

Our AI will be explained in depth in the paragraph 9.6.

5.2.5. Revising Application Interfaces (Sprint 4)

In the fourth sprint, views and user interfaces of the older version will be revised and upgraded. The target we seek, it is to renew the look and feel of the application and improve the final user experience.

In this sprint we have to analyze old interfaces and regard its weak points or interactions that can be improve. Then, we have to design new interfaces and interactions that avoid this mistakes. Next, we will implement this new interfaces and finally, we will test and make ready to deliver this increment of software.

This sprint went from 20/03/18 to 10/04/18. And we covered the user stories #3,#4,#5,#6,#8,#9,#10,#12 and #13.

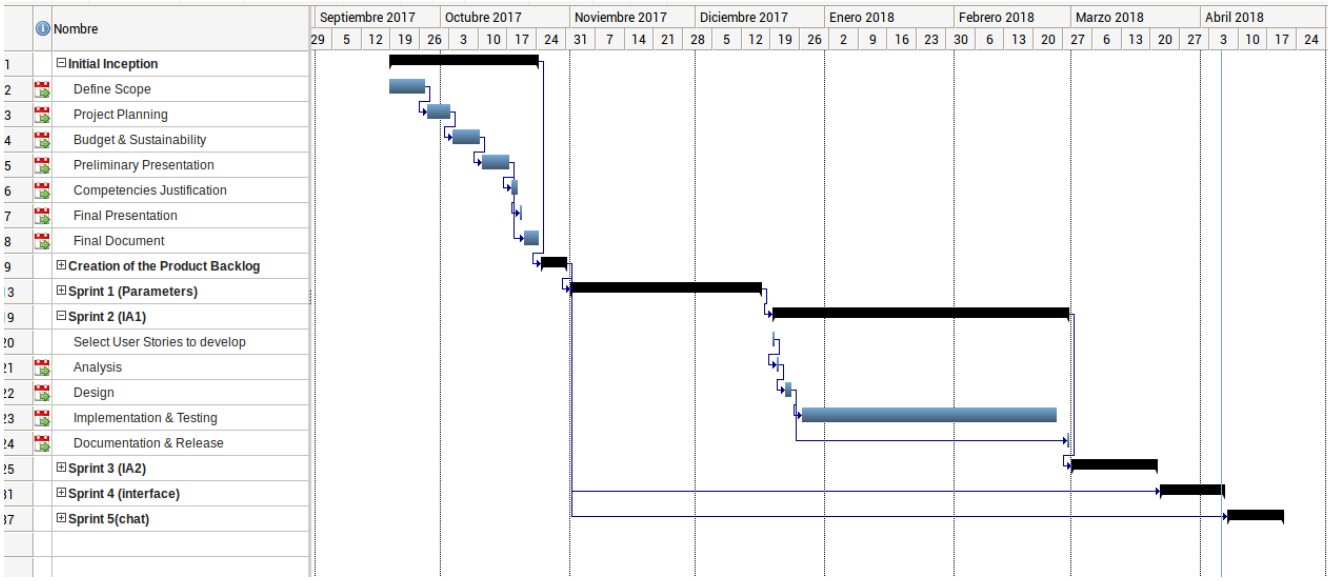
5.2.6. Changing the game chat (Sprint 5)

The last sprint was meant to develop the application chat. Digging in to the application, we realized there already was a chat so we decided to make some changes to make it more accessible. This sprint was shorter than the rest going form 11/04/18 to 18/04/18 and we covered the user stories #7 and #11.

The change applied to the interfaces and the chat are going to be explained in the chapter 9.3.

Sprints were meant to be of a 3 week duration. Each sprint will be considered as a 100% functional software release Although we will try to finish every task in its sprint, it is possible that user stories are finished before or after the sprint deadline. For this reason, Scrum recommends that in the beginning of every sprint it must be a planning of the user stories to be done during that sprint.

5.3. Gantt Chart



* Sprint 3, 4 and 5 maintain the same structure of the rest of sprints

5.4. Alternatives And Action Plan

Scrum allows some tolerance to obstacles and incidentals because at the end of each sprint it is mandatory to do a meeting in order to evaluate the project evolution. If a problem is detected or there is a delay on the release or, simply, priorities are modified, in the meeting at the beginning of the sprint actions will be taken in order to solve the problem in this iteration. Also, if we got run out of time for the final delivery we will have to reduce the scope or maybe extend the calendar.

5.5. Resources

The available resources to make this project are:

- **Acer Aspire A5**: Development and documentation hardware. It is the main tool of development because all the project is going to be developed in it.

- **Huawei P8 Lite**: Testing hardware. It is the device used to test how the application works in Android.

- **Ipad 9’7”**: Testing hardware. It is the device used to test how the application works in iOS.

- **Visual Studio Code**: Development software. It is the IDE we are going to use to develop the application. It is free.

- **Google drive and LibreOffice**: Documentation software. They are the tools we are going to use to generate most of the application documentation.

- **Gantter**: Documentation software. It is the tool we used to generate the Gantt diagram.

- **Trello**: Scheduling software. It is the tool we used to track the tasks throughout the whole development.

- **Android studio**: Development software. We need android studio in order to build the apk for Android devices.

- **Xcode**: Development software. We need xcode in order to generate the .app file for iOS devices.

- **MacBook Pro**: Building machine. In order to run xcode we need a machine that has installed the macOS.

6. Budget

The budget of this project will be calculated on the basis of the activities exposed in the Gantt diagram of the previous section.

6.1. Cost Identification and Estimation

6.1.1. Direct Costs

Project direct costs are derived of the needed workforce in order to develop it. The main roles that will affect the project and their wage are:

Product Owner: He is the responsible to transfer the project vision to the team, create and manage the product backlog. He is paid a salary of 19 €/hour.

Scrum Master: He makes the team to follow processes and rules of the scrum methodology. He tries to reduce project obstacles and he also works with the product owner in order to prioritize the product backlog. He is paid a salary of 15 €/hour.

Multidisciplinary developers team: Professionals with the needed technical knowledge that will develop the project. They are paid a salary of 13 €/hour.

Activity	Estimated Hours	Role	Cost
Define Scope	56	Product Owner	1064
Project Planning	32	Product Owner (60%) & Scrum Master (40%)	556.8
Budget & Sustainability	40	Product Owner	760
Preliminary Presentation	40	Product Owner	760
Competencies justification	16	Product Owner	304
Final Presentation	8	Product Owner	152
Final Document	16	Product Owner (90%) & Scrum Master (10%)	297.6
Declare User Stories	8	Product Owner (50%) & Scrum Master (50%)	136
Refine User Stories	16	Product Owner (50%) & Scrum Master (50%)	272
Create Test Validation	16	Product Owner (50%) & Scrum Master (50%)	272
Select User Stories To Develop	20	Developer (80%) & Scrum Master (20%)	268
Analysis	40	Developer	520
Design	64	Developer	832
Implementation & Testing	480	Developer	6240
Documentation & Release	56	Developer (90%) & Scrum Master (10%)	739.2
Total Project Cost *	908		13173.6

Tab 1. Direct Costs

**We considered journeys of 8 hours per day but we know that, for TFG, it is calculated to have journeys of 3,5 or 4.*

6.1.2 Indirect Costs

Indirect Costs that we are going to assume are:

- Electricity: The average energy consumption of a laptop is 0.11 kWh. Given the project duration (908 hours), the laptop will consume 93.17 kWh. The kWh price in Spain is 0.12442€. The result is an expense of 11.59€.

Contrarily, the price of the tablet and cellphone energy consumption will be assumed of 0€ because their use will not only restrict to this project and they will be working anyway.

- Software: It is foreseen that we will only use free software.

- Hardware: The price of the laptop used for developing was 350€ and it has a useful life of 5 years. The cellphone price was 288€ and it has a useful life of 3 years. In the tablet case, the amortization is null because it is a temporal grant. In addition, it will be used a free server to run the application.

- Telecommunications: The ADSL price is 47 €/month. Due to the project we are using it at least ⅓ of the day (8 hours). So we calculate the cost is 15.66 €/month.

Concept	Price per Unit	Time	Dedication Percentage	Estimated Cost
Electricity	0.124 €/Kwh	93.17 kWh	100%	11.5922114
Software	0 €/month	7 months	100%	0
Hardware				
Laptop	5.83 €/month	7 months	100%	40.81
Cellphone	8 €/month	7 months	10%	5.6
Tablet	0 €/month	7 months	10%	0
Telecommunications	47 €/month	7 months	33.33%	109.6666667
Total Cost				167.67

Tab 2. Indirect Costs

6.1.3. Contingencies and incidentals

Setting a 15% of contingencies, we get:

Concept	Cost	Contingency	Total
Direct Cost	13173.6	1976.04	15149.64
Indirect Cost	167.67	25.16	192.83
Total			15342.47

Tab 3. Total + Contingencies

The more important incidental and the ones that are going to affect the budget are:

1.- A delay on the releases that delays the finishing of the overall project for 2 weeks.

2.- A fault on the laptop that makes us to buy another one.

Incidental	Probability	Price	Total
14 day Delay	20%	1,040€	208€
Laptop replacement	10%	350€	35€
Total			243€

Tab 4. Incidental Prevision

Finally, the resulting budget is 15,585.47 €.

6.2. Management Control

It will be an evolution control from the conclusions originated in the meetings at the end of each sprint. In these meetings, there will be an analysis of the project state by monitoring the compliance of the deadlines and modifying the budget if necessary.

7. Sustainability

	PPP	Product Life	Risks
Environmental	Design Consumption	Ecological Footprint	Environmental Risks
	8	9	-5
Economical	Bill	Viability Plan	Economical Risks
	6	8	-10
Social	Personal Impact	Social Impact	Social Risks
	9	4	-3
Sustainability Range	23	21	-18
	26		

7.1. Environmental

This project will be developed on a laptop which means a lesser energy consumption than desktop computer consumption. The ecological footprint of the project will not be very large due to the fact that the running server, which the application will be deployed, will be working anyways and a few more processes will not make a difference in terms of CO2 emissions. The main environmental risks are that the game becomes massively played and the energy consumption shoots up.

7.2. Economical

The biggest consumption of resources will be done while we are developing the project given the requirement of workforce needed. During the useful lifetime, the project only needs the application server running which drops down the necessary capital to maintain the application. The risk of the project revolves around the delay of the releases that make the budget grow.

7.3. Social

The project will occupy a great space in the mind of the developer during a considerable time frame and it is expected to provide him a reasonably good knowledge for his future. Referring to the audience, which this project aims to, it is not expected a really big impact (and it should not) because we are developing a game after all. The way could have a bigger impact is that the game allows to the players to meet other people that can share interest with them. The biggest social risk is that the game goes viral and it becomes an important source of distraction especially with the youngest.

8. Specification

8.1. User Stories

Here are listed the user stories that our system has to accomplish:

User Story	1
As a	User
I want	To create parameterizable games
So that	I can play games with different constraints and play-styles
Priority	High
Conditions of satisfaction	You can play with every combination of parameters. Parameters doesn’t enter in contradiction.
Length	64 units

User Story	2
As a	User
I want	An AI to play against
So that	I can practice or simply play alone
Priority	High
Conditions of satisfaction	AI can make a move. AI can pass a turn.
Length	128 units

We realized that this user story was too long so we divided it in two.

User Story	2.1
As a	Developer
I want	That the application gather more in-game information
So that	The AI can have a better control of the game
Priority	High
Conditions of satisfaction	The AI has access with all the data needed
Length	64 units

User Story	2.2
As a	User
I want	An 3-difficulty AI to play against
So that	I can practice or simply play alone
Priority	High
Conditions of satisfaction	AI can make a move. AI can pass a turn.
Length	64 units

User Story	3
As a	User
I want	To save my session
So that	I don’t have to login again when I leave the application
Priority	Medium
Conditions of satisfaction	User can go out an return and his session stays open.
Length	16 units

User Story	4
As a	User
I want	To log out of the application
So that	Another user can use his users in the same device
Priority	Medium
Conditions of satisfaction	User can log out User can log in afterwards The opened session is destroyed
Length	16 units

User Story	5
As a	User
I want	To know which are the parameters of the games
So that	I can adapt my play-style
Priority	Medium
Conditions of satisfaction	Parameters are displayed in the lobby view
Length	4 units

User Story	6
As a	User
I want	To know the result of the games I have played
So that	I know who won
Priority	Medium
Conditions of satisfaction	The result of the game is shown to the user
Length	4 units

User Story	7
As a	User
I want	To chat with other users without having to pass through the game
So that	I can chat with them in a easier way
Priority	Medium
Conditions of satisfaction	There is a tab in the initial view that allows the user to access to his chats
Length	16 units

User Story	8
As a	User
I want	To stay in the game even if I have played my turn
So that	I can prepare my next movement
Priority	Medium
Conditions of satisfaction	The player can access to the game when it isn't his turn. The player can stay in the game when it isn't his turn.
Length	32 units

User Story	9
As a	User
I want	To know who's turn is
So that	I know if I can do my movement or not
Priority	Low
Conditions of satisfaction	The system displays who's turn is
Length	2 units

User Story	10
As a	User
I want	To save my favorite parameters of the game
So that	I can replay a similar game
Priority	Low
Conditions of satisfaction	The user has the possibility of saving the game parameters. The user has the possibility of loading the game parameters.
Length	8 units

User Story	11
As a	User
I want	That my display name is unique
So that	I know who a play with
Priority	Medium
Conditions of satisfaction	The system validates that the display name it isn’t used when a user does the registration
Length	16 units

User Story	12
As a	User
I want	To know if a chip is correctly placed
So that	I can play accordingly
Priority	Low
Conditions of satisfaction	The system validates chips when placed The system shows incorrect chips
Length	8 units

User Story	13
As a	User
I want	To change the chips of my hand position
So that	I can prepare a move
Priority	Low
Conditions of satisfaction	The system allows to change chips in hand positions
Length	8 units

Length units are powers of 2 and although they doesn't represent a real value, they help us to make an idea about the length of the user stories.

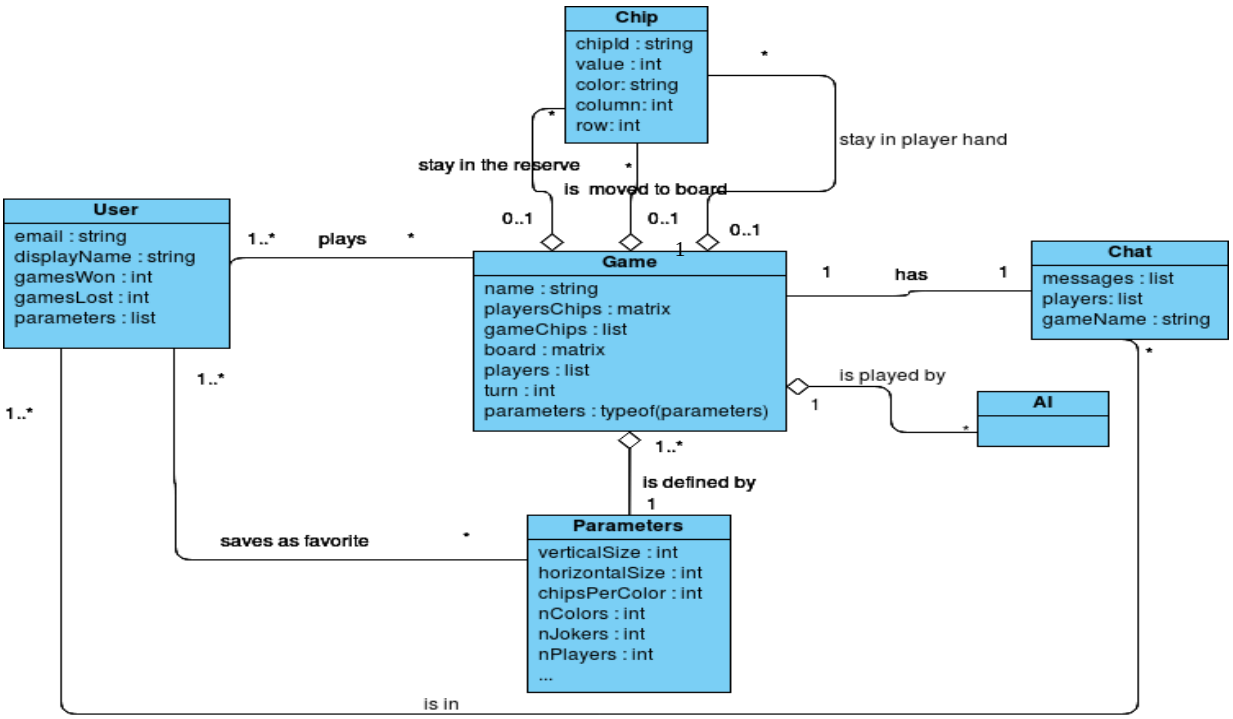
Obviously, the system has many more user stories and use cases but we aren't including them in this enumeration because they are already developed in the previous version of the game so we think that it's better to focus on what we are going to do in this version. For more information, consult the specification paragraph in the older version project report [15].

This user stories will be developed in different sprints:

- Sprint 1: User story #1
- Sprint 2: User story #2.1
- Sprint 3: User story #2.2
- Sprint 4: User stories #3,#4,#5,#6,#8,#9,#10,#12 and #13
- Sprint 5: User stories #7 and #11.

8.2. Conceptual model of the system

Here is the conceptual model of the system:



The users can play all the games they want. Games can have from 2 to 4 players with a chat for them. Chips can be in the board, on the player's hand or into the game heap of chips. Chips have a color, a value, even the jokers, and a identifier that allows to distinct the repeated ones. Users have their email and a unique display name. Also the will have the amount of games won and lost for ranking purposes. In this version, the chats will stay available after the game ended. We will talk about the game parameters and attributes later. For reasons of space, only a part of the parameters is represented but we mean that they are all there.

8.3. Old version analysis

In this chapter we are going to do an in-depth analysis about the older version of the application by explaining errors, limitations and possible changes. We are going to do this analysis by checking all the features view by view.

8.3.1. Login

The login view is correct, the user can log in or to choose if he want to create an account.

The main problem we detected with the login is that it doesn't remember the user. Every time the user opens the application he has to fill the form. We are maintaining the view like it is in this version.

8.3.2. Create a user

This view is correct too, user can create an account without losing track of what he's doing.

The problem detected here is that although two users can't have the same email address because Firebase manage it, however they can have the same display name. Also, once the user has done the registration process, he has to pass through the login process again. We are maintaining the view like it is in this version.

8.3.3. Lobby

In this view the user can enter to an active game or create a new one. In this view we detected some problems:

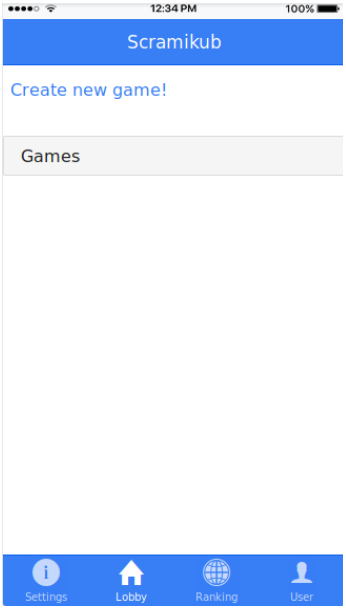
First, when the user enters to this view and the user has some games, the games take some time to appear. While this is happening the user doesn't know what happened to his games and could be confused some times.

Second, the mails of the users are shown for each game, not making use of the display name we have asked to the user. By showing the mails, users could see their privacy violated so we think it is better to show the display name.

Third, users don’t know what happens with their games when they end. There is no warning no visual interaction with them, they just disappear from his sight.

Forth, that is highly related to the third, the user has no clue of how many games he have played: there is no history.

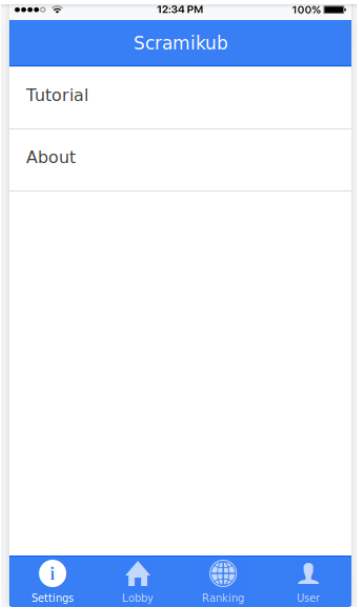
Fifth, the user can’t open a game when it is not his turn so he can’t prepare a movement.



Although the main structure of the view will stay the same some changes must be done in order to fix this problems.

8.3.4. Settings view

This view only has two buttons one for the tutorial and one that show us information about the app. The about button works as pretended but the tutorial one doesn't do anything, it is just a blank button. We need to get rid of this button.

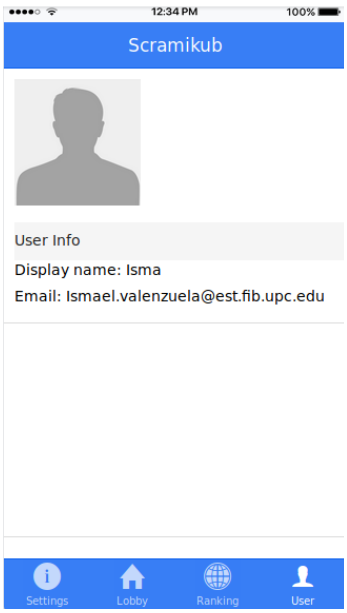


8.3.5. Ranking

This view is correct, it works well and do what it has to do. There is only a complain against it and it is that even though it shows the users with a best score (gamesWon – gamesLost), the application doesn't registers that games won and lost. They only can be modified by manipulating the database. We are maintaining the view like it is in this version.

8.3.6. User information

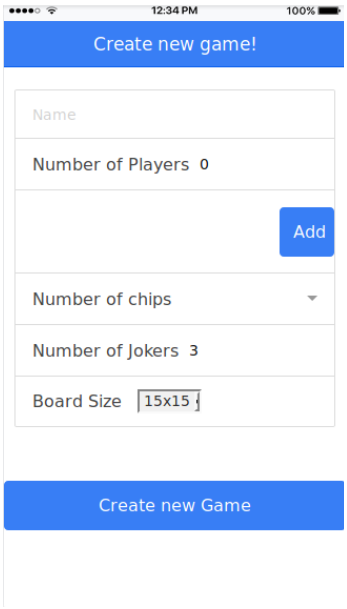
The only thing we miss of this view is the information about the games (gamesWon, gamesLost). It could be a propagation of the error highlighted on the ranking paragraph. We are maintaining the view like it is in this version just we will add the game information.



8.3.7. Create game

In this view the games are created. The user can choose the game name and the player against who is going to play. More options are shown to the user but they are blank. The process only works with hard-coded parameters. Also, the process doesn't check if the game name is already used or checks that the final number of players and the number of players chosen are the same.

We want to highlight that the second time that we tried to create a game we couldn't make it. This process, obviously, needs and will have an upgrade.



8.3.8. Select Players

This view works perfectly. All the users are shown an the user has to choose as much as number of players he want for this game. The user creating the game is automatically marked as a player. We only will try to speed up the process of choosing rivals by randomizing them if the player wants.

8.3.9. Games

The game view shows the board, the player chips and 5 buttons: Previous, Next, Pass, Play, Return Chips. Previous and Next are used for navigating through the user's hand pages. Pass shows a warning pop up and proceeds to pass turn. Play commits the chip movements and Return Chips is meant to return the moved chips to the user's hand. In this view we have found some things that could be upgraded. The board is scrollable and it has programmed a zoom in and a zoom out that works fine.

First things first, we think that the label "Return Chips" is confusing.

Second, when a user does a group in the board, he doesn't knows if that group is correct or not: the game does not give a clue of what is happening.

Third, when the user wants to move a chip from his hand, he drags it and drops it into the board; but when he wants to change a the position of a chip on the board he has to tap on it and then tap where he wants to move it, this leads to user confusion and frustration.

Forth, the button return chips returns the moved chips by reloading the page which takes some time and this process could be quicker.

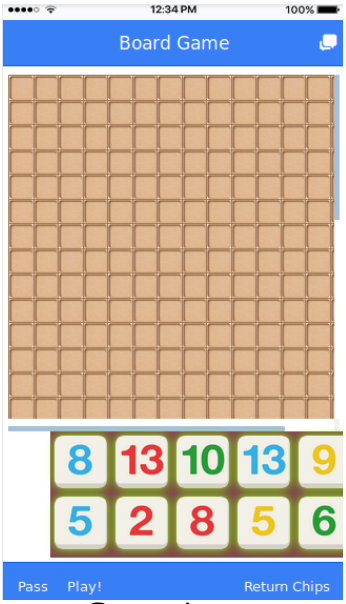
Fifth, the user can't change the positions of the chips in his hand. This way the user can't organize his hand properly.

Sixth, if the board has an incorrect chip, the player can't commit the move so there is no way that the users are penalized with the incorrect chips of the board.

Seventh, Pass and Play redirect us to the lobby instead of letting the players prepare the next movement.

Eighth, as stated before, when a game ends there is no warning for the players the single player that is warn is the one who won the game that, by the way, already knows that he won.

Ninth, depending of the screen resolution the buttons Previous and Next disappear.



Game view

8.3.10. Chats

The chat view shows the messages sent and who has sent them and a little form to introduce a new message.

First of all, we don't approve the display of the user's mail as we noted earlier, so this need to change in order to show his display name. Also, when the user enters to the chat he doesn't know who is chatting with if nobody said anything. We need to display all the chat participants. Another thing we want to highlight is that if the user want to send a quick message to a friend he has to open the game both are playing (that is only available if it isn't finished), wait that the chips and board loads and then open the chat.

9. System design

9.1. Technology used

To develop this project we used Ionic Framework + Firebase. Ionic is an angular.js based framework that allows us to develop and deploy HTML5 applications in mobile devices. Ionic uses WebViews to imitate the behavior of a native application both Android and iOS. Also, it have incorporated predefined controls and directives that make the development quicker and emulate native controls.

About Firebase, we only used two of all the services that it offers. We use:

- Real Time Database: This service allows to have our database and our application data synchronize. It has a Javascript API that allows us to integrate it with angular.js. This database is NOSQL and works with JSON documents.

- Firebase Auth: This service that creates and authenticate users. As the real time database, it has a Javascript API.

9.2. System architecture

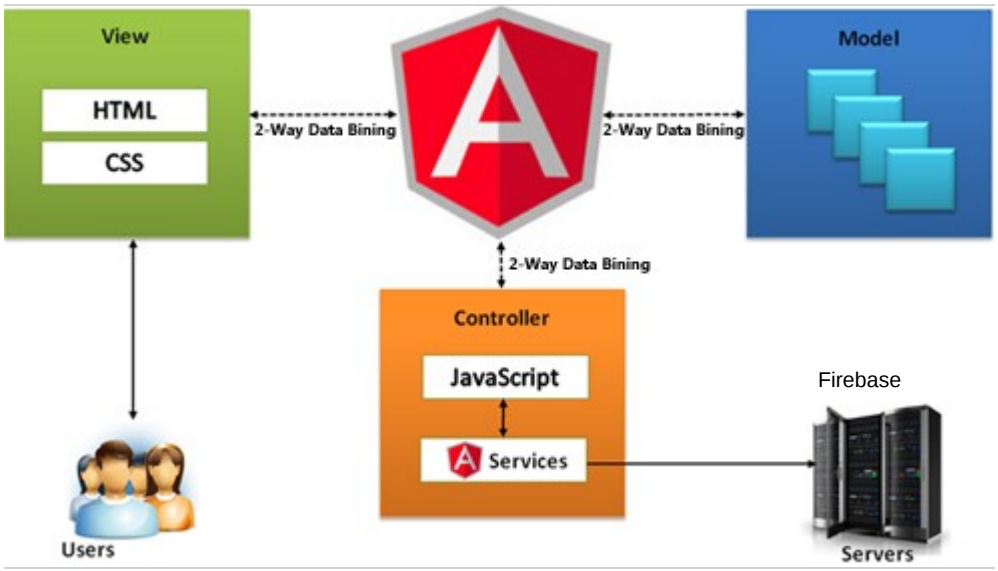
We considerate that the architecture of the older version of the application continues to fit our need so we think it is better to maintain it. Once said this, our application architecture is a client-server architecture, but the majority of the work is done on the client-side. The server only has two functions, authenticate users and storing all the data that our app needs and generates. On the client-side, it is where all the code of the application is including our AI module.

The design pattern used it's the well-known MVC pattern. This pattern consist in three components:

- Model: It's the responsible of the data of our application. It is the lowest layer of the app.

- View: It's the responsible of showing the data to the users and capturing user events. It is the highest layer of the app.

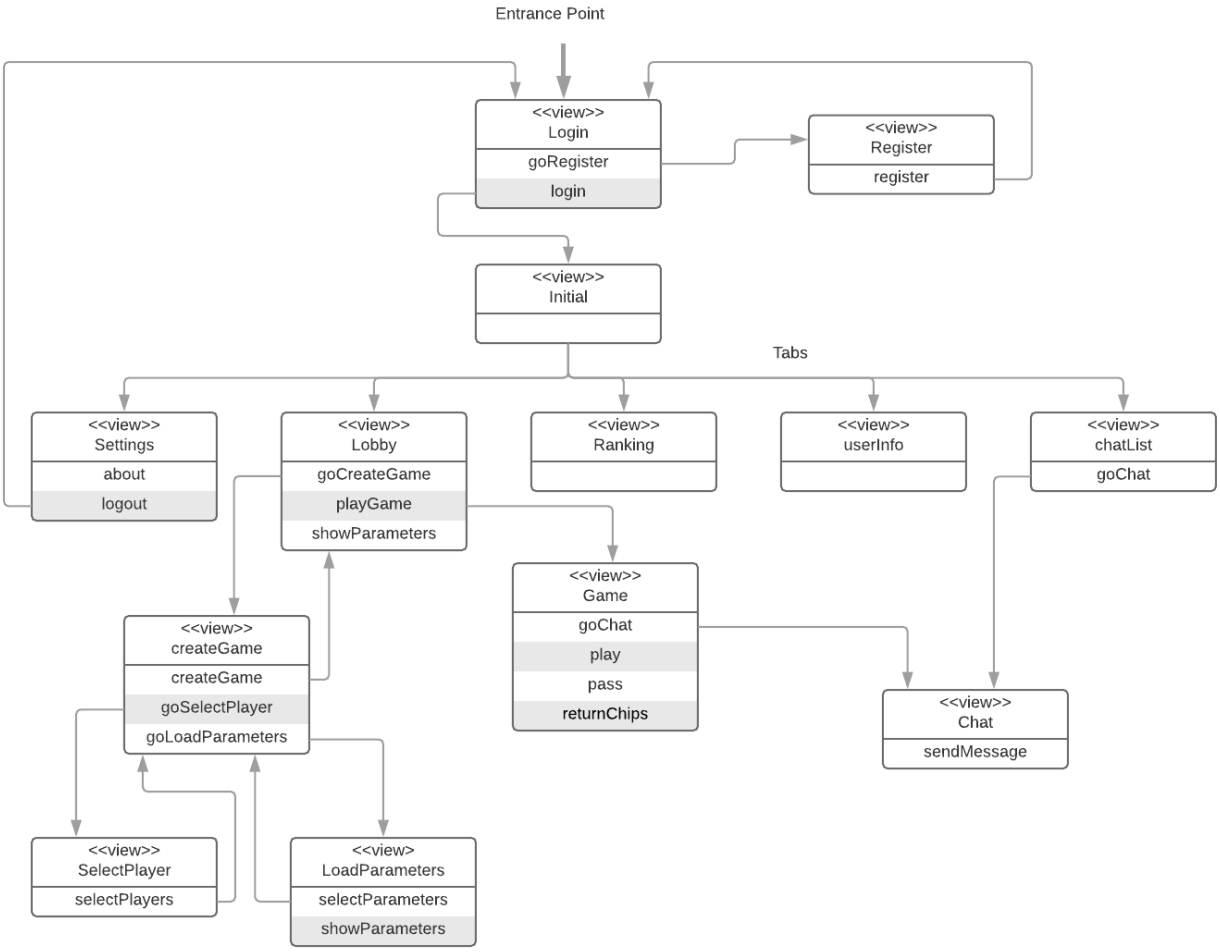
- Controller: It's the responsible of filling the view with the pertinent data and reacting to the user events. It is the middle layer.



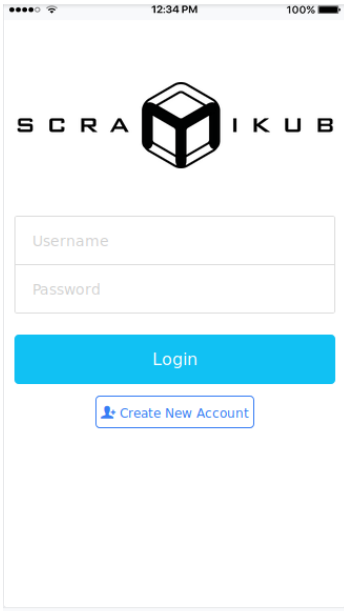
The schema above show us how angular.js integrates with our system. Angular uses a 2-way data binding in order to have the data synchronized between all of the components. Also, we use angular services to interact with Firebase.

9.3. Navigational Map and User Interfaces

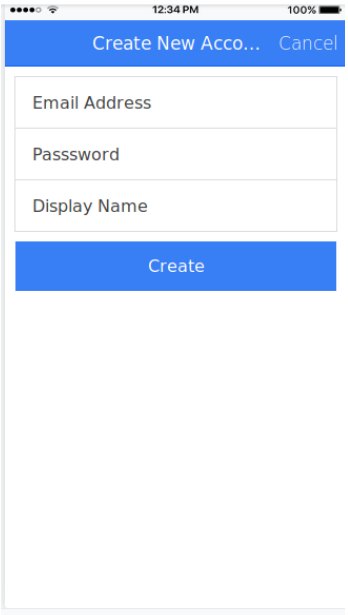
In this paragraph we will expose the navigation map and the user interfaces of our app. Here it is the navigation map of the application:



The user enters directly to the screen of login. If he isn't a registered user, he can got to the the register form. If the user was authenticated before, it will be a variable in the local storage of the app that will make him to login automatically.

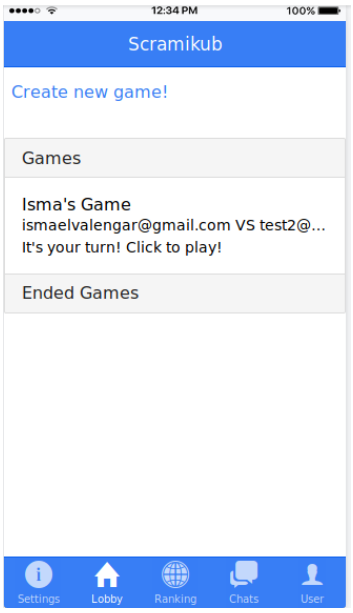


Login view

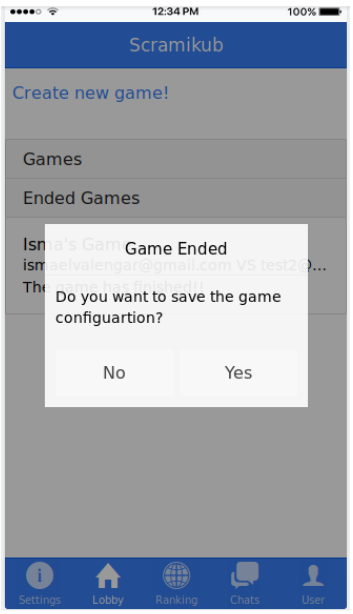


Create User View

Once he has done the login, we show him the lobby tab of the initial view. In this tab, we show him all the games he's in and the option of creating a new game. He can navigate to the other tabs too. If a game ends when the user is on this page a pop up will ask him if he want to save the configuration of this game. Also, when the user holds a tap on a game, we show a pop up with the parameters of that game.

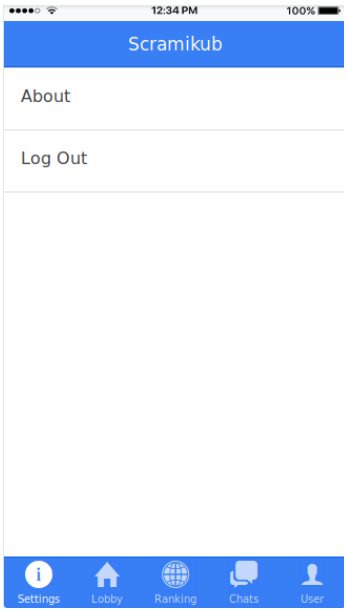


Lobby view



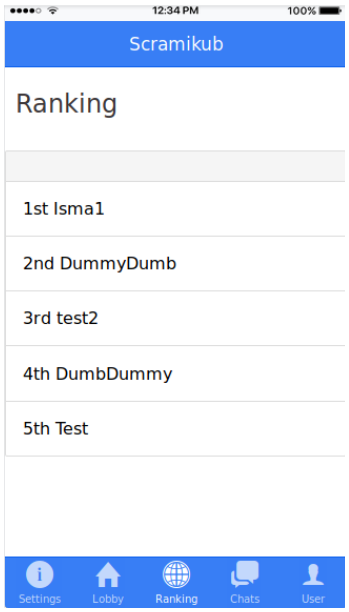
Pop up

The settings tab is designed in order to be upgraded in future iterations of the application. In this version it is only used for logging out and showing information about the app. Logging out the application leads us to the logging view again.



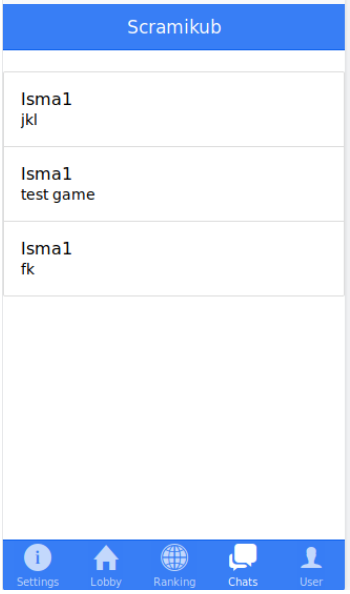
Settings view

The ranking tab show us the top five players with the highest scores of the game.

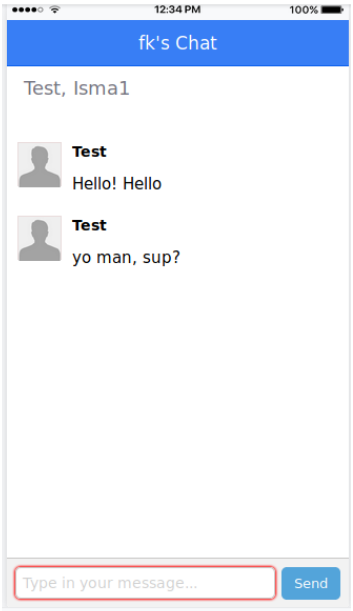


Ranking view

The chatlist tab has a list of the chats that the user has opened. By tapping on a chat, the chat view is opened.

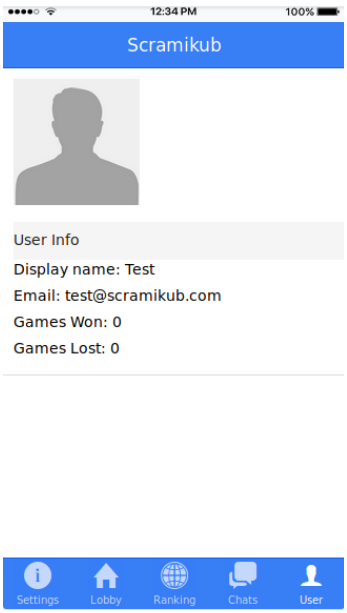


ChatList view



Chat view

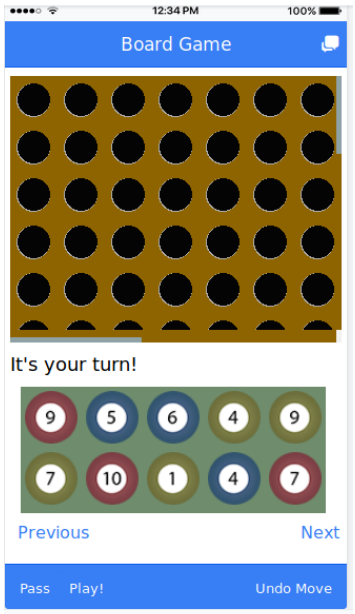
The last tab is the user view. The user view show the information about the player.



User view

The game view shows the board, the user's hand and six buttons. The button on right-top corner opens the chat of that game that is the view that we already discussed. In the middle of the view we show the board that is where the user can place his chips. Down the board, we have a textbox that show whose turn is. Then, we display the chips of the user also known as the user hand. By clicking on them, he can choose where to them: onto the board or in another space of his hand. The buttons Previous and Next allow the user to navigate through his hand. The three buttons on the bottom are for passing the turn, confirming the move made and to undo the movement.

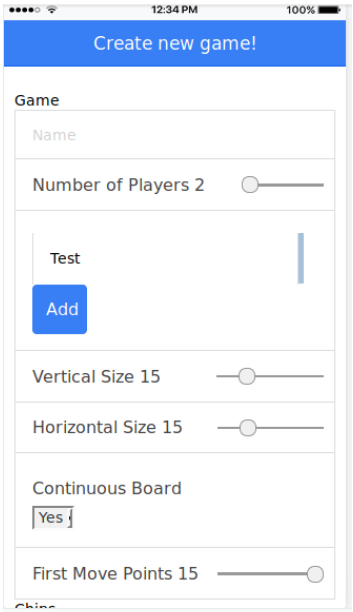
When it's not the user turn, chips can't be placed on the board and the bottom buttons are disabled so that the user is not able to do a move.



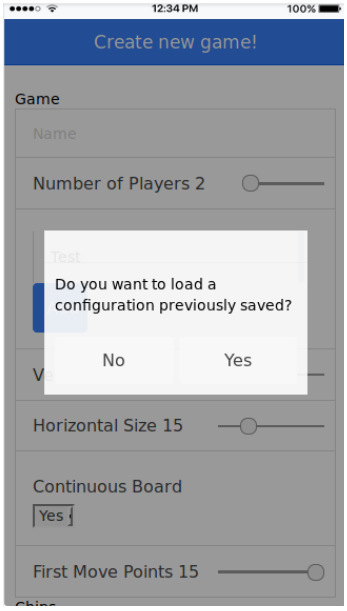
Game view

As the image shows, int his version, we decided to use circular chips and cells because on the original board game they are circular too.

The create game view is meant that the players can choose the parameters they want to their game. If the user has parameters saved, we will inform him with a pop up that he can load a configuration. If he decides to load it, the application will open the loadParameters view. Parameters can be defined by moving the sliders or for some we need to select a option from a listbox. When the user clicks add, the view changes to the selectPlayers view. Once the player has chosen the parameters he want, he can tap to create game to finish creating the game.

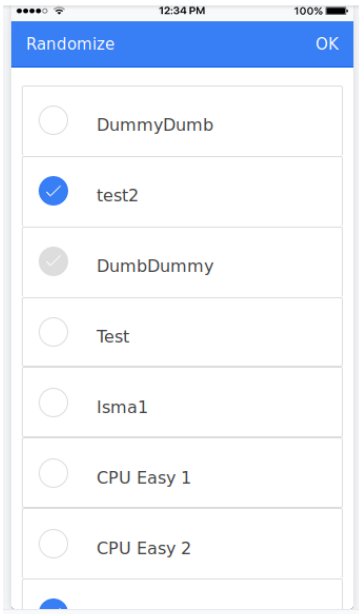


CreateGame view



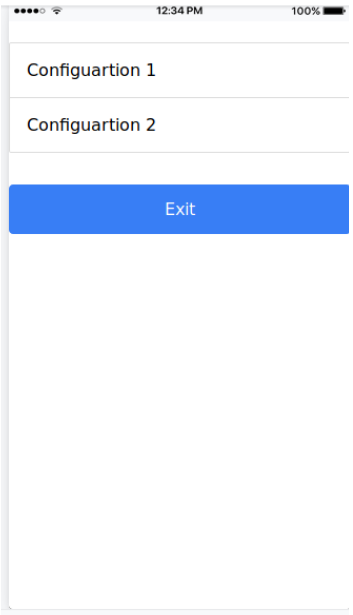
Pop up

Select Player view is simple. The user just has to chose the player he wants to play against and tap OK when he is finished. As we can see the player who is creating the game is already selected. We also added a randomize button that chooses the players randomly.

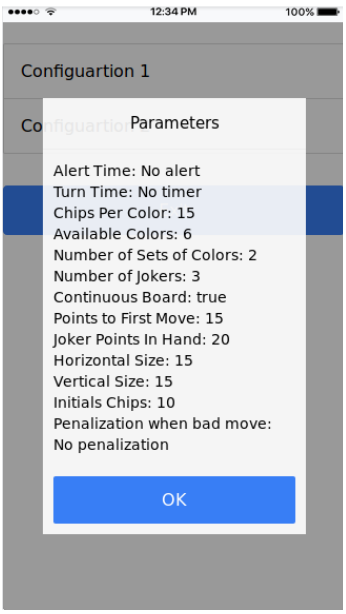


SelectPlayer view

And finally, the loadParameters view show the player what game configurations has saved. By holding the tap on a configuration, it shows which are the parameters of that configuration. Tapping on a configuration will load it and returns us to the create game view.



LoadParameters view



Parameters of a config.

9.4. Parameter description

In this paragraph we will expose the game parameters and how they interact between them.

9.4.1. Game Parameters

These are parameters that define the board and general aspects of the game.

- Number of players. This defines how many players can play the game. It can be a number from 2 to 4. In the original game, games were form 2 to 6 players but we thought that is very difficult for the user of an application to follow a game with more than 4 players because to many changes can happen and also it could get the users frustrated because the turn wait time would be so long. It, also, tell us the minimal number of chips that the game must have because the minimal number of initial chips in hand is 3 for each user.
- Vertical and horizontal sizes. These describe the sizes of the board. It could go from 7 to 40. They define the maximum number of chips the game can have, so they constraint the parameters of chips per color, number of colors and number of sets.
- Continuous Board. This determines if the board is circular: edges of the board are contiguous. It does not affect other parameters.
- First Move Points. This determines how many points must have the first group of chips to be placed in the board by each player. It goes from 0 to 15.

9.4.2. Chip Parameters

These are parameters that refer the chips of the game.

- Chips per color. This determines the chips that a color will have. It goes from 1 to 30. It restrict the number of colors and the number of sets that a color can have. Also, it affects initial chips because it is one of the 3 parameters that defines how many chips the will have.
- Number of Colors. This determines the colors the game will have. It can go from 1 to 10 and the colors available are blue, red, yellow, purple, green, black, brown, pink and sky blue. It restrict the number of sets and as chips per color does, it affects initial chips.

- Number of Sets. This parameter deter mines the number of repetitions a color will have. It can go from 1 to x, depending the maximum chips this game can have. As the 2 above parameters, it also affects the initial chips.
- Number of Jokers. This defines the number of jokers a game will have. It goes from 1 to 10. This parameter is not affected by any parameter.
- Initial Chips. This is the parameter that defines how many chips the user will have on his hand when the game starts. If the number is low the game will probably be shorter than if the number is high. It goes from 3 to x, where x is the total chips of a game divided by the number of players.

9.4.3. Other Parameters

- Bad Move Penalization. This parameter determines which is the penalization when a player does an incorrect move. By the moment, the values it can take are: no penalization (the player is penalized with a chip more) and extra chip (the player is penalized with an extra chip plus the default one).
- Points for a Joker in Hand. This is the points a joker has when it in the hand. It goes from 1 to 0 and it’s not affected by any other parameter.

9.4.4. Parameter combinations

The recommended parameters of the game are the one our application loads as default that are the parameters that the original rules dictate. This are: 2 players, 15 x 15 continuous board, 15 points to do the first move, 15 chips each color, 6 different colors repeated with 2 sets, 3 jokers, 15 initial chips on the hand of the player, no extra penalization and 20 points per joker in hand. Also, for a quicker but yet pretty average games, we recommend to change to 10 jokers, which is the maximum; only 3 colors, 10 chips each color and 10 initial chips. This way we increase the chances of getting more chips that are able to form a group.

Not allowed combinations happen when the amount of the chips available couldn’t fit inside the board so the game could not be finished. Our system takes care of this by limiting the user inputs. An example of a not allowed combination could a 15 x 15 board with 30 chips per color, 6 colors, and 3 sets because the amount of chip spaces is 225 (15x15) and the amount of chips is 540 (30x6x3).

9.5. Database design

In this version of the application, we kept the older database schema but we added some new attributes. Our database is a NOSQL document based database. Firebase uses a JSON object to register all the data. Our database is divided in three major branches:

JSON document

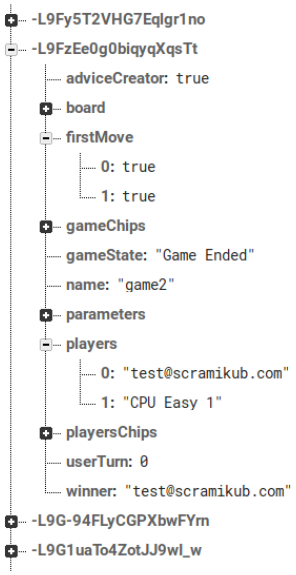
9.5.1. Chats structure

Chats contains all the data of the chats of the app. Every chat is registered with an identifier that Firebase provides. Each one has a game name which indicates from what game comes, an array of the messages sent and the users that are involved on the chat. Every message has the text of the message and the email of the sender. Users are also saved by the email. This structure stays the same of the old version.

Chat structure

9.5.2. Games structure

Games contains the data about the games. Games are also identified with a Firebase identifier (which in the example is L9FzEe0…) . Game can have an adviceCreator attribute which is used only in order of warn the players that the game ended in case they want to save the parameters used in this game. The board attribute saves the data of the board. It is a matrix that registers what is in every box of the board. It will be deeper explained after. The firstMove attribute is an array of booleans that tell us if the users have done the first move because we have to control if he passes the FirstMovePoints parameter. GameChips are the chips that neither of the players have and are not in the board. GameState is an internal string that show what is the state of the game. Name is the name of the game that can't be repeated. Parameters has a collection with the parameters explained before. Players is an array of the players' emails. Player chips is a matrix which every row are the player chips. Userturn is an integer which indicates whose turn is. And winner is the email of the player who has won which is empty while the game isn't finished. Games that haven't end also have a groups attribute that tell us which are the complete groups on the board.



Game structure

This is the most enlarged structure. Of all of the attributes of this structure in this version we added the adviceCreator array, the firstMove array, the groups array and the collection of parameters of this game.

9.5.3. Users structure

Users contains the data about the users. Just like games, every user has an identifier given by Firebase. Users have a displayName that it what the other players see, an email in order to be authenticated an two integers, gamesWon and gamesLost, that register the amount of games lost and won. If the user has decided to save some configuration, the application create an array called parameters that registers every configuration he had saved. In our version we added the parameters array, the rest of the structure stays the same.

Users structure

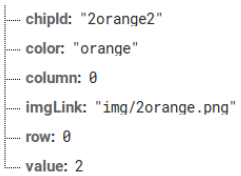
9.6. Artificial Intelligence Design

The artificial intelligence development is an important part of our project. In this paragraph, we are going to explain which data needs to make a move, which changes of the move evaluation we made in order to gather that data and which process the AI follows when it is its turn.

9.6.1. Knowledge Base

In order to make a decision our AI needs to extract specific data from the game. Our knowledge base gathers both actual data about the chips (what chips are on board, what chips the AI has, ...) and meta-data of the game (how many groups are in the board, which is the size of the board, ...). We'll show now this data on detail:

- Chips: every chips on the game have this structure, even empty ones:



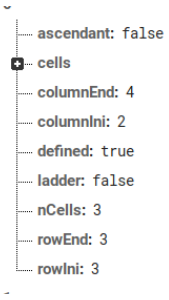
  ChipId is a unique key for every chip on the game, it identifies the chip. Color and value are the color and the value that chip has. In the case of jokers, the color will have the value "joker" and for the value it will take -1. This value is called undefined value. Column and row are the color and row of the board where the chip is. These two attributes can seem redundant, there are cases were we have no access to the matrix and still want to know in which column and row the chip is. Last, we have imgLink that is the relative URI of the image that represents the chip. The value of this attributes for an empty chip is an empty string ('').

- Board: Our AI needs to have access to the board. Every game has a board matrix called board. Every pair (row,column) of the matrix represents a row and a column of the real board. In every pair, there is a chip or an empty chip.

- CPU chips: The chips the CPU has are in a row of the matrix playersChips. Each row of the matrix playersChips is the hand of chips a player has. The chips on playersChips are semi-full because every attribute of them is filled with the respective values except for the row and column. We could give an advantage to the AI if we let it to access other player's chips, but we think it will be unfair and could become unsatisfactory for the players.

- Parameters: Our AI need to know the configuration of the game. Specially: the parameters that refer to the size of the board (verticalSize and horizontalSize) and the continuity of the board (continuousBoard) in order to know where it could put a group of chips, points of the first move (firstMovePoints) in order to know if that play could be made or not in the early states of the game; and the number of chips a color has (chipsPerColor) in order to know how groups can be made.

- Groups: In order to speed up the process of the AI our AI uses the groups attribute. This attribute is created at the end of the turn where a correct group that also passes the firstMovePoints is introduced. Groups is an array that gathers data from all the correct groups that are in the board. A group follows this structure:



  Cells contains the chips that establish the group. RowIni and columnIni are the row and column where the group start. RowEnd and columnEnd are the row and column where the group end. The groups will be always registered left to right or up to down so the unique cases where the row/columnIni will be greater than row/columnEnd is when the board is continuous and the group are on the edges. These four attributes apart from indicates us where the group begins and ends, also it show us if a group is vertical or horizontal because we know if columnIni is the same as columnEnd that group shares the same column so the group is vertical. Same thing happens with rowIni and rowEnd. Ncells is the number of cells, this number always is going to be bigger than 3 because of a group needs 3 or more chips to be completed. Defined, ladder and ascendant are 3 booleans that show us what kind of group is. Defined tells us if the group is defined. A group is defined when two or more have a real value. A group can has more than three chips and still not be defined because of the jokers. If defined is false, ladder and ascendant are irrelevant. Ladder shows us if the group is a ladder or a set of chips with the same value and different color. If ladder is false, ascendant is irrelevant too. Finally, Ascendant tells us if the ladder is ascendant or not.

9.6.2. Move Evaluation

The changes of the move evaluation aim to two goals. One, making the evaluation process more efficient and two, filling all the data needed for the AI. For the purpose of understanding the changes of the evaluation process we are going to explain how the old process was.

The old process analyzed the entire board every time the user made a move and it registered if it was correct or not. If the board was correct you could commit the moves by tapping play. If the board was incorrect, the player can't commit the move so here is were an interesting part of the game was lost. Because the player looses the chance of getting the incorrect chips of groups he have destroyed. In our opinion this is an interesting mechanic of the original game that adds a deeper level of complexity.

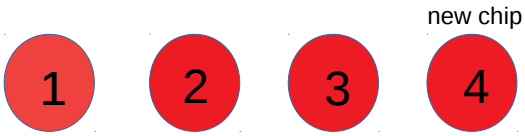
Now, our process analyze only the chip moved and its surroundings. First, we have to differentiate the possible movements a player can do:

- A player inserts a chip from his hand.

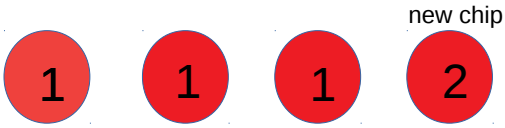
- A player changes the position of a chip in the board.

When the player insert a chip from his hand, we take it from his chips and place it on the board. Now, our evaluation process begins. First, it checks if the chips is compatible with the new neighbor chips. Then, it takes this 4 neighbors (up, down, left, right) and depending of the number of non-empty neighbor chips, it will proceed:

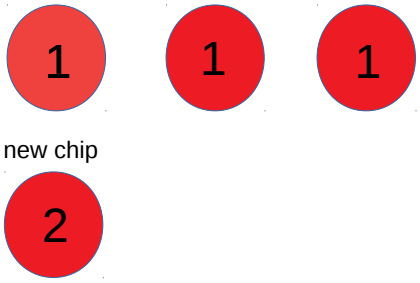
- If there's only 1 neighbor: it will check if the chip can be added to one of the two possible groups (one vertical group and one horizontal) that the neighbor chip can be. This leads us to 3 different scenarios: the chip can be added to a group so it is added; the chip can't be added to any of the 2 groups so the chip will be classified as incorrect and we create an incomplete group just for it; and the last scenario, the neighbor chip only is in a group but the new chip doesn't follow the direction of this group so the chip is also classified as incorrect but a new group with 2 chips (this chip and the neighbor one) is created.



This is the first scenario, new chip is compatible and can be added to the group.



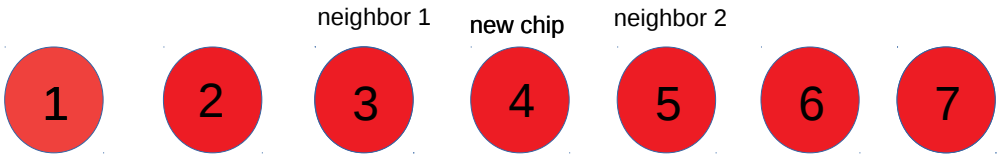
This is the second scenario, new chip although being compatible it can't be added to the group because it doesn't fit the kind of group.



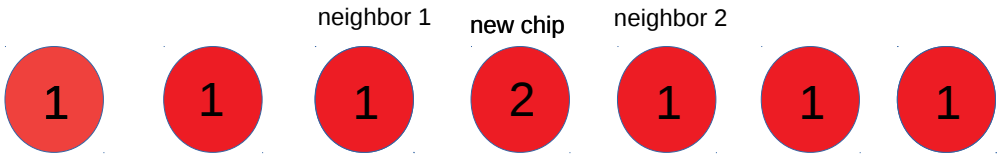
This is the third scenario, new chip is compatible but it can't be added to the group because it doesn't follow its direction.

- if there are 2 neighbors: the process will proceed depending of the position of this neighbors:

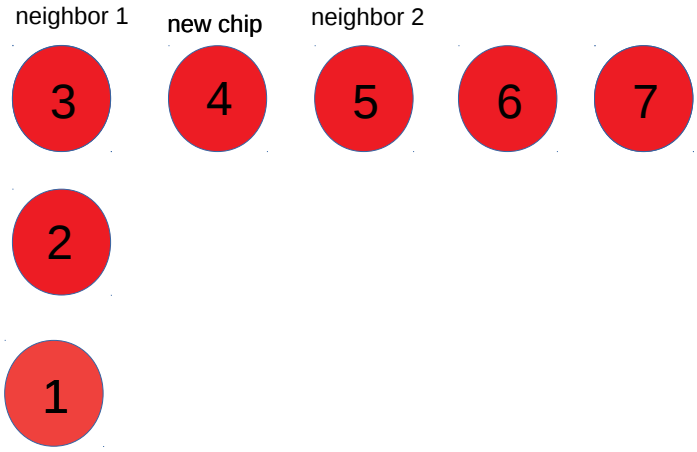
  - if the neighbors share the same direction (up-down or left-right) the process will check if the chip can be added to both groups. Now this leads us to 4 scenarios: first scenario, chip can be added to a group for each of the neighbors so the chip is added to one and the 2 groups are merged; second scenario, chip can't be added to any group for each neighbor so the chip will be classified as incorrect and we create an incomplete group just for it, third scenario chip can be added to a group of a neighbor but the other neighbor chip only is in a single group and the new chip doesn't follow the direction of this group so both chips the new chip and the single grouped are added to the group of the other neighbor; and four scenario, both neighbors have a single group each and the new chip doesn't follow their direction so a new group with the three is created.



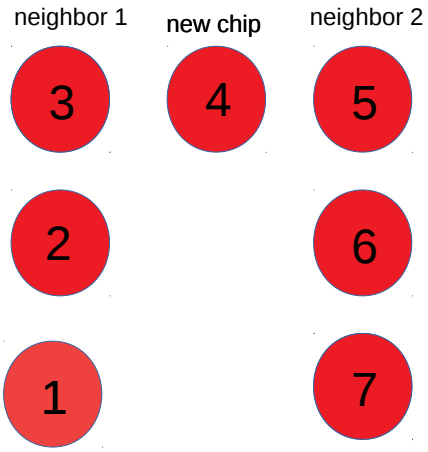
First scenario



Second scenario

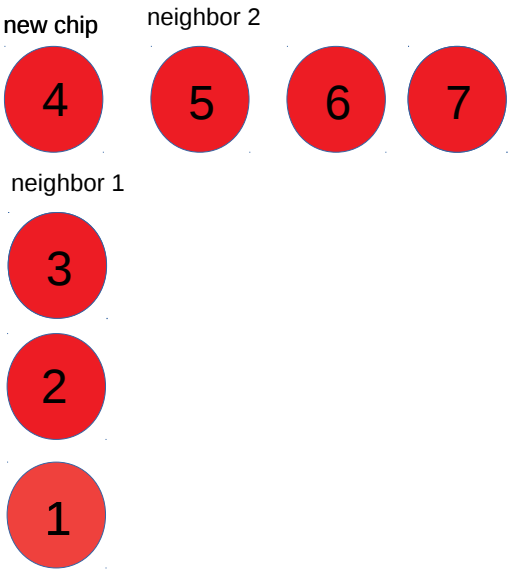


Third scenario

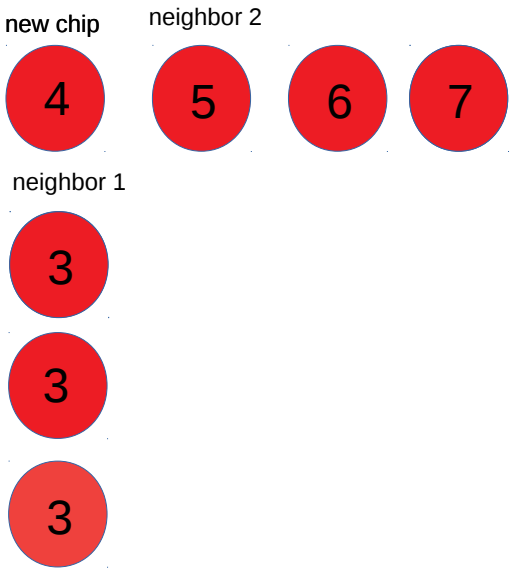


Forth scenario

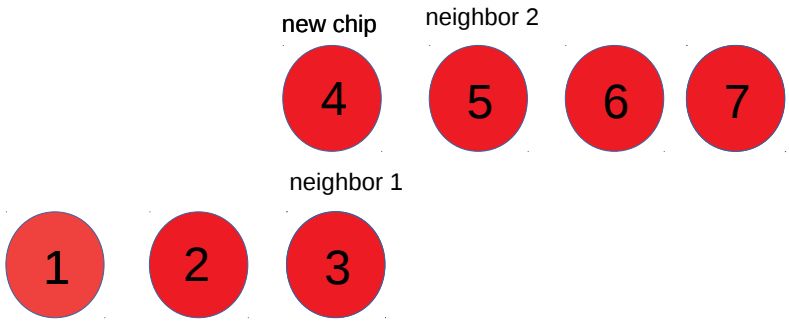
- if the neighbors don’t share the same direction the process will pass for the same 4 scenarios but it will act similar to the case when there is only a neighbor: in the first scenario, this time the chip will be added to both groups only; in the second scenario, the process stays the same; in the third, the chip will be added to the group and it is created a new group for the chip and the other neighbor; and in the forth scenario, two groups will be created one for the new chip and a neighbor and the other, for the new chip and the other neighbor.



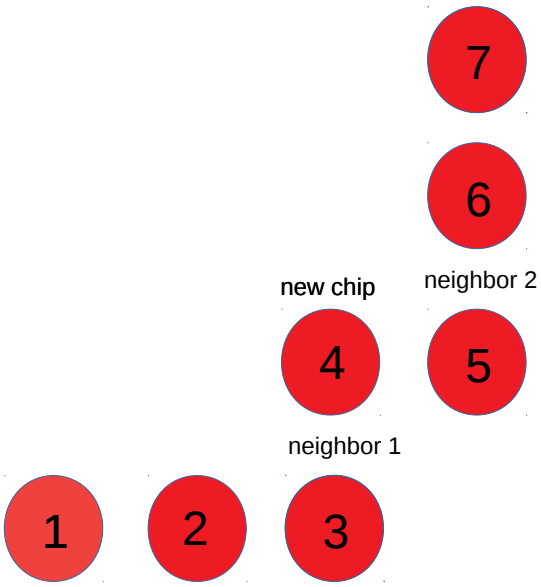
First scenario.



Second scenario.



Third scenario.



Forth scenario.

- If there's 3 neighbors: the process will do a mix of the processes follow in the 2 neighbors in the same direction and the case of a single neighbor.

- And finally if there's 4 neighbors: the process will do two processes of the case of 2 neighbors in the same direction.

We want to highlight, that when a chip is detected as incorrect, it won't be added to any correct group. It will just be added to a unitary group and be classified as incorrect. Also, we want to clarify why this unitary or uncompleted groups are created. They are created because our process needs groups to evaluate in which situation the chip is. When the process detects that the number of cells of a group is bigger than 3 this group will become correct.

Now, we are going to discuss the other action the user can do: changing a chip position. When a user changes the position of a chip, two things can happen to the groups that chip is depending of the position of the chip:

- If the chip is in the beginning or the end, the group is updated (rowIni or End/ columnIni or End) and it can become incomplete and undefined.

- If the chip is in the middle, the group is split in two so there are two new groups that have to be analyzed by the process in order to know which is their state.

Remember that this process has to be executed two times because the chip can be in two groups altogether. Then, when the chip is extracted from the groups, the chip can be considered as a chip in hand and it will pass the whole process of inserting a chip from a hand in its new position.

Finally, we are going to remark the added difficulty that a joker brings. Jokers add uncertainty when we try to make groups. When a joker is inserted to a group the joker sometimes won't have a concrete value: it could has two or more values and the move will still correct. This leads a to the case that when a undefined (has no concrete value) joker belongs to two different undefined groups. What happens? When one of the two groups becomes defined the joker acquires a value and that value could not be allowed by the other group. Also, every time the user changes a joker to another position, the joker becomes undefined again. We need to keep this in mind when we work with jokers.

9.6.3. AI Logic

Our artificial intelligence is meant to have three behaviors one for each difficulty.

- The easy AI tries to do the best move with the chips on his hand. To do so, it will do a loop while the best move generated has less chips than 3 or there is no space in the board. Then, it will do another loop for each chip in his hand, catching each as a root. Each loop will try to make an ascendant ladder, a descendant ladder or a repetition group with the rest of the chips taking the root chip as the first chip of the group. Also, each loop will update the best move, which is initialize with 0 chips, with the group with more chips out of the three that we attempted to create earlier. Once all the chips in the hand are processed, if the number of chips of the best move is greater than 3, the best move will be done and our list of chips will be shorter. Now, if we still in the main loop, we are going to do the same loop as before but now with the new list of chips because maybe now we have another possible group in our hand. This way we only leave the main loop if our best move is an incorrect group (nCells < 3). We want to note that in a new version this difficulty can be changed by another less challenging that only make the first group it can.
- The medium AI tries to do the best move on his hand and also adds the chips to complete groups on the board. To do so it follows the same process of the easy difficulty and when it has done the best move, it proceeds to add some chips to the complete groups. To extend complete groups, we do a loop for every chip of our hand and we try to add it on the beginning of a group if it is not possible we try to add it in the end. Note that this can also be modified to make an outer loop that loops while any chip in added but we agreed to fix this level of difficulty as medium.
- The hard AI tries to do something similar to what the easy does. This AI consider all the chips that are on game are in its hand taking advantage of the rule that says that when a chip is incorrect and the player ends the turn that incorrect chips go to the player hand. The process adds all the chips to the AI hand and then it follows the same process as the easy AI. This difficulty is constantly changing the board so the player gets confused and it makes the game more challenging.

All this AIs share the same process for doing a move. Doing a move will receive a move and it will place it on the board. In order to place it on the board, it will try to get consecutive group of empty cells and add the cells there. This consecutive group of cells is found with a process that choose a cell of board randomly and check if the consecutive cells are empty.

Once the AI has done finished of doing all the moves it can it will update the database and pass the turn to the next player.

10. Development

In this paragraph, we are going to describe the new features and explain the fixes that we had done.

10.1. Login

As we used the new version of the firebase API we needed to change to way that our users authenticate. Also, here is where we save in the localStorge the user data in order to satisfy the user story #3.

```
$scope.login = function(user) {
    if (user && user.email && user.password) {
        $ionicLoading.show({
            template: 'Signing In...'
        });
        firebase.auth().signInWithEmailAndPassword(user.email, user.password).then(function (authData) {
            console.log("Logged in as:" + authData.uid);
            ref.child("users").child(authData.uid).once('value', function (snapshot) {
                var val = snapshot.val();
                $scope.$apply(function () {
                    $rootScope.userInfo = val;
                });
            });
            $ionicLoading.hide();
            userConnected = authData;
            firebase.auth().setPersistence(firebase.auth.Auth.Persistence.LOCAL);
            localStorage.setItem("authData",JSON.stringify(authData));
            $state.go('initial');
        }).catch(function (error) {
            alert("Authentication failed:" + error.message);
            $ionicLoading.hide();
        });
    } else {
        alert("Please enter email and password both");
    }
};
```

Login function

10.2. Creating a user

For this feature, we added two validations. The first will validate that any user name or email begins with CPU because we need that names for the AI players and the second will validate that every displayname is unique. Also, we updated the Firebase API calls and added a new behavior that consists in login the new user once it is created.

```
firebase.auth().createUserWithEmailAndPassword(user.email,user.password).then(function (userData)
        var usrRecord = {
            email: user.email,
            displayName: user.displayname,
            gamesWon: 0,
            gamesLost: 0
        };
        ref.child("users").child(userData.uid).set(usrRecord);
        console.log(usrRecord);
        $ionicLoading.hide();
        $scope.modal.hide();
        firebase.auth().signOut();
        $scope.login(user);
}).catch(function (error) {
    var alertPopup = $ionicPopup.alert({
        title: 'Error',
        template: error
    });
    $ionicLoading.hide();
```

Some create user code

10.3. About

This feature only shows a pop up to the user with the names of the developers. It basically the same of the older version plus the developer of this version.

```
$scope.about = function() {
    console.log("about");
    $scope.data = {};

    var myPopup = $ionicPopup.show({
        template: '<img src="img/title.png" style="width: 100% !important;"/> Oriol Burgaya <br/> Ismael
        title: 'About',
        subTitle: '',
        buttons: [
            { text: 'Ok' },
        ]
    })
}
```

10.4. Logout

The logout feature signs out the Firebase session, remove the localStorage added in the login and returns to the login view.

```
$scope.signOut = function() {
    firebase.auth().signOut().then(function() {
        localStorage.removeItem("authData");
        document.location.href="/";
    }, function(error) {
        console.error('Sign Out Error', error);
    });
};
```

Logout function

10.5. Entering to Play the Game

In this feature we select the game the user has selected and go to the game view. We took the turn restriction so the players can enter the game even though if it's not their turn.

```
$scope.clickGame = function(game){
    var actualGame = angular.toJson({ "actualGame": game});
    $state.go('game', { 'actualGame': actualGame });
};
```

Click game function

10.6. Showing parameters

When the player holds a game tap, we detect it with the angular attribute on-hold on the html template and we execute the showParameters function which pops up the configuration of the game in a way the user can understand.

```
$scope.showParameters = function(game){
    if(game.parameters == null) return;
    var s = "";
    var params = game.parameters;
    var aT = params.alertTime == 0?"No alert":params.alertTime;
    var tT = params.turnTime == 0?"No timer":params.turnTime;
    s+= "Alert Time: " + aT + "<br/>";
    s+= "Turn Time: " + tT + "<br/>";
    s+= "Chips Per Color: " + params.chipsColor + "<br/>";
    s+= "Available Colors: " + params.nColors + "<br/>";
    s+= "Number of Sets of Colors: " + params.nRepetitions + "<br/>";
    s+= "Number of Jokers: " + params.nJokers + "<br/>";
    s+= "Continuous Board: " + params.continuousBoard + "<br/>";
    s+= "Points to First Move: " + params.firstMovePoints + "<br/>";
    s+= "Joker Points In Hand: " + params.handJokerPoints + "<br/>";
    s+= "Horizontal Size: " + params.horizontalSize + "<br/>";
    s+= "Vertical Size: " + params.verticalSize + "<br/>";
    s+= "Initials Chips: " + params.initialChips + "<br/>";
    s+= "Penalization when bad move: ";
    if(params.badMovePen == "no_pen")
    s+="No penalization";
    else if(params.badMovePen == "extra_chip")
    s+="Extra Chip";
    else
    s+="Loose Turn";
    s+="<br/>";

    var alertPopup = $ionicPopup.alert({
        title: game.name + " parameters",
        template: s
    });
};
```

ShowParameters function

10.7. Saving Configuration

When the system detects that a game has ended with the function $scope.games.$watch() from angular, we show a pop up to advice the player if he want to save that game configuration. This feature, we update the advice creator attribute of the game and, if the user response is affirmative, add the configuration to the user parameter array.

```
var ind2 = j;
var myPopup = $ionicPopup.show({
    template: 'Do you want to save the game configuartion?',
    title: 'Game Ended',
    buttons: [
        {
            text: 'No',
            onTap: function(e) {
                console.log(ind2);
                game.adviceCreator[ind2] = true;
                Games.$ref().child(game.$id).child('adviceCreator').set(angular.copy(game.adviceCreator));
            }
        },
        {
            text: 'Yes',
            onTap: function(e) {
                game.adviceCreator[ind2] = true;
                Games.$ref().child(game.$id).child('adviceCreator').set(angular.copy(game.adviceCreator));
                var params;
                if($rootScope.userInfo.parameters != null){
                    params = $rootScope.userInfo.parameters;
                }
                else{
                    params = [];
                }
                params.push(game.parameters);
                Users.$ref().child($scope.userConnected.uid).update({'parameters':params});
            }
        }
    ]
});
```

Pop up code

10.8. Ranking

Although the ranking view and controller worked fine. We had to add some code in the game controller so when a game ends, it registers if the user actually won the game or lost it. Here is the code added:

```
for(var i = 0; i < $scope.actualGame.players.length; i++){
    var playa = getUserByEmail($scope.actualGame.players[i]);
    if(playa != null) {
        if(playa.email == winner){
            playa.gamesWon++;
        }
        else{
            playa.gamesLost++;
        }
        updateUser(playa);
    }
}
```

10.9. Listing chats

We list all chats the user has opened. By clicking the chat we open the chat view.

```
$scope.clickChat = function(chat){
    var g;
    for(var i = 0; i < $scope.games.length; i++){
        if($scope.games[i].name == chat.gameName){
            g = $scope.games[i];
            break;
        }
    }
    if(g != null){
        var actualGame = angular.toJson({ "actualGame": g});
        $state.go('chat', { 'actualGame': actualGame });
    }
    else{
        var alertPopup = $ionicPopup.alert({
            title: "Internal Error",
            template: "Game doesn't exist"
        });
    }
};
```

10.10. Displaying user information

We added the amount games lost and won so the user can see how he is doing. We only gather the info of the user and display it in the view.

10.11. Loading user's configurations

When the system detects that the user has any configuration saved, it alert him to load one if he wants. With $stateParam.asked we know if we already asked him during the game creation process.

```
if($rootScope.userInfo.parameters != null && !$stateParams.asked){
    var myPopup = $ionicPopup.show({
        template: 'Do you want to load a configuration previously saved?',
        buttons: [
            {
                text: 'No',
                onTap: function(e) {
                }
            },
            {
                text: 'Yes',
                onTap: function(e) {
                    $state.go('select-params', { 'user': angular.toJson($rootScope.userInfo) });
                }
            }
        ]
    });
}
```

Load user pop up code

10.12. Selecting Players

This feature still the same but we added 9 CPU players (3 each difficulty) so the player can choose to play against the AI. The CPU players are a pseudo-user that has a fake email and a displayname that show its difficulty. The alreadyAdded boolean is to control that we don't add the CPU players each time the player enters to the view.

```
var cpuHard3 = {
    email: "CPU Hard 3",
    displayName: "CPU Hard 3"};

var alreadyAdded = false;
for(var i = 0; i < $scope.users.length; i++){
    if($scope.users[i].email == "CPU Easy 1"){
        alreadyAdded = true;
    }
}

if(!alreadyAdded){
    $scope.users.push(cpuEasy1);
    $scope.users.push(cpuEasy2);
    $scope.users.push(cpuEasy3);
    $scope.users.push(cpuMedium1);
    $scope.users.push(cpuMedium2);
    $scope.users.push(cpuMedium3);
    $scope.users.push(cpuHard1);
    $scope.users.push(cpuHard2);
    $scope.users.push(cpuHard3);
}
```

A CPU pseudo-user and the code that add them to the complete list of users

10.13. Creating a Game

This features has changed a lot because of the introduction of the parameters. Every time the user changes a parameter, we have to make sure that the game can be played and finished so our code has a lot of checking. For example: when a size of the board is changed, we need to change the chips per color so all the chips can fit in the board. This creates a chain reaction because when the chips per color are changed, the number of colors have to change they make change the number of sets that change the number of initial chips.

```
$scope.sizeModified = function(){
    $scope.auxiliarData.maxChips = Number($scope.data.verticalSize) * Number($scope.data.horizontalSize);

    if ($scope.auxiliarData.maxChips >= 30){
        $scope.auxiliarData.maxChipsColor = 30;
    }
    else{
        $scope.auxiliarData.maxChipsColor = $scope.auxiliarData.maxChips;
    }
    $scope.auxiliarData.minChipsColor = 1;

    $scope.data.chipsColor = Math.trunc(($scope.auxiliarData.maxChipsColor + $scope.auxiliarData.minChips

    $scope.chipsColorModified();
}
```

A function of the chain reaction

In the view we use sliders because this way it's easier to us to control the user input. When a slider change, it triggers a function depending of which parameter it is. The function above is called when the size changes.

Finally when the user hits the create game button we add the new game to our games and also create another chat.

10.14. Chatting

The chat feature keeps the same structure of the old version. The major changes we have done are updating the accesses to Firebase to the new API and adding a display that shows who are in the chat. Also, we don't display emails anymore just display names.

New Firebase access

10.15. Playing the game

The playing the game is the biggest and the core of our application. The first change we did was to change the evaluation process. This new process executes almost entirely when the user moves a chip to the board even when it is already in the board. With a complex logic, our process will add the chip to a group or not or classify it as incorrect. When the chip is not compatible with another chip, it will be not moved and it will stay where it is. Here is it some pseudo-code of the new process:

```
Function checkForGroups(chip,row,column){
	chip.row = row;
	chip.column = column;
	cellsAround = getCellsAround(chip);
	for each cell in cellsAround{
		if(!areCellsCompatible(cell,chip))
			undoChipMove();
			end;
	}
	numberNeighbours = countNeighbours(cellsAround);
	if(numberNeighbours == 0) chipIsIncorrect(chip);
	else{
		if(attemptToAddToGroups(cellsAround,numberNeighbours,chip))
			group = addChipToGroup(chip);
		else
			chipIsIncorrect(chip);
	}
}
```

We also took the drag and drop objects because they weren't acting like supposed and the users got confused. Now all the interactions are done by tapping. We also now allow the user to stay in the game when he ends his turns. In the older version, we load everything when the users enters to the view now we have to keep track of the movements that other users do while he is this view.

In order to give some visual feedback to the users we added an $interval which executes every 500 ms that illuminates the chips that are incorrect.

```
var off = true;
var timer = $interval(function (){
    for(var i = 0; i < $scope.incorrectChips.length; i++){
        var table = document.getElementById("table-board");
        var chip = $scope.incorrectChips[i];
        var cell = table.rows[chip.row].cells[chip.column];

        if(off){
            angular.element(cell).addClass("incorrect-cell");
        }
        else{
            angular.element(cell).removeClass("incorrect-cell");
        }
    }
    if(off) off = false;
    else off = true;
},500);
```

$interval function

10.16. AI Players

The AI has 3 levels of difficulty. In the first level, the AI only will move complete groups from its hand. On the second level, the Ai will do like the first but also put chips in the board complete groups. On the last level, the AI will reorganize all the chips it can (board chips and hand chips) in order to get the best move possible. The AI player's structure is similar in every case. Here it is the easy one structure:

```
function cpuEasyMove(){
    var usrTurn = $scope.actualGame.userTurn;
    $scope.movedPoints = 0;
    organizeHand();
    if(moveGroupsHandEasy()){
        console.log("move made");
        if ($scope.actualGame.playersChips[usrTurn].length == 0 &&
        $scope.incorrectChips.length == 0 && (!$scope.actualGame.firstMove[usrTurn]
        || ($scope.actualGame.firstMove[usrTurn] &&
            $scope.movedPoints >= $scope.actualGame.parameters.firstMovePoints))) {
            saveActualGameWithWinner();
        } else {
            saveActualGame();
        }
    }
    else{
        console.log("move not made");
        cpuPassMove();
    }
}
```

organizeHand puts all the jokers on the end of the hand in order to be our last option. MoveGroupsHandEasy make the move that will expend more chips. This move could be a group or several. If the move can be made, it will save the movement as a normal move or as a win if it is the case or it will pass the turn. Just to mention it, both functions name after saveActualGame are the same that we use to save the real players.

In consequence of introducing the AI players, every time a turn user turn ends we validate if is now time to the CPU to make a move and prepare the move if it is the case.

11. Obtained Results

In this paragraph, we are going to discuss the results obtained with the project and the different obstacles.

In the project we have developed a parametrization of the games while its design we realized that it wasn't possible to make all the parameters variables and even though there were some that could be they must be limited so the games could be played and finished. One example of this is the relation board size – total chips that we already explained. The most arguable decision taken about the rules of the game is that we have decided to allow the users use repeated colors in the same group as long as they are not consecutive. This decision was taken in order to ease to the users to create new groups. Also, there are parameters we couldn't end to develop, like the introduction of a turn timer or the extension of the incorrect move penalizations, because we run out of time during the sprint.

During the development of the AI, we realized that with the actual process of checking the chips movements, we couldn't develop a good AI because we needed more information of what was happening in the board. In order to get around this new circumstance, we needed to make a rearrange of the sprints because it was a biggest obstacle than what we initially thought. Now, our system updates the board information we need every time the user changes a chip position.

The next step was to create the AI. We made three difficulties (easy, medium and hard). The first two difficulties are programmed similarly but the second is a bit more complex. The hard difficulty uses a different algorithm. In the first instance, the hard difficult was meant to be more complex and challenging to the user but we accomplished to made it fun enough so the users can enjoy playing against it.

Then, we had to focus in the user interactions and interfaces. We remove the confusing interactions like the way the drag and drops worked, added new interactions and views to make the user experience richer and comfortable. Also, we had to fix some issues with some controllers that weren't working properly.

We made some fixes to the chats. We added a new tab to the initial view so the player can enter directly to the chat they want so we minimize the number of taps needed. Also, we don't show user emails anymore so we preserve the user's privacy.

Finally, we want to expose the problems we found when we attempted to test the application in the two operative systems. The android build worked well and there was no problem for testing it into the physical device. The problems found for this device were related to some css clauses that were incompatible with the Android Webview and some redirections weren't working properly. When working with iOS, we weren't able to test it in

a real device because Apple demands us to have a Developer account which is paid. We just were able to test it inside the Xcode simulator. The main problem, beside that, found in iOS is that the operative system doesn’t provide a back button so we need to implement a back button for every view.

12. Conclusions

In this paragraph we will expose the conclusions taken while making this project as well as the degree of accomplishment of the goals and some future upgrades.

The biggest conclusion taken is that sometimes is hard to understand how other people develop. Every one has his own style and good habits and bad habits and in some complex parts is difficult to follow where the other people flow of thoughts is going.

As to the project's goals, we have managed to meet them in a high grade. During the project we have made a parameterization of the games, we have created a AI that plays against the users using three different behaviors and we changed the most confusing user interactions and we have remodeled views to make them more convenient. Also, to emphasize, we had to fix some old bugs we found while developing.

The technical competence that we emphasized are the one that refers to the developing and maintaining a distributed application because our project is basically that a distributed application network supported. Regarding the technical competence of Identifying, evaluating and managing potential risks**,** we haven't gone too deep, honestly. We identified the potential risks and tried to avoid them but we didn't expend too much time to manage them. About defining and managing the software requirements and designing appropriate solutions in one or more domains have been work in a similar level and they are decently accomplished.

As we commented in the chapter 5.1, during the development of the project, the developer was unable to work for the project because of his other job. He had some unexpected workloads that kept him away of the project and it was impossible to make any progress. Also, we found more loose ends and things that weren't properly working on the old version. For this two reasons we were forced to extend the duration of the project for another three months.

The future work lines could pass by upgrading the AI providing it a more complex behavior including implementing a machine learning algorithm and finish the implementing of turn timers. We think that making possible to publish our app on Google Play and Apple Store or making a port to windows phone could be interesting and also, making a revision of the artistic section of the game (insert some music and sounds, using more complex color palettes, creating animation,...).

13. References

[1] *The Global Games Market report* Newzoo

https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/

[2] *Market Brief — 2017 Digital Games & Interactive Media Year in Review* Superdata

https://www.superdataresearch.com/market-data/market-brief-year-in-review/

[3] *Mobile Gaming Statistics* MediaKix

http://mediakix.com/2017/05/mobile-gaming-statistics-gaming-apps/#gs.tgsNfWo

[4] *Scrabble Application* Google Play and App Store

https://play.google.com/store/apps/details?id=com.ea.scrabblefree_na

https://itunes.apple.com/us/app/scrabble/id501724085?mt=8

[5] *Apalabrados* Google Play and App Store

https://play.google.com/store/apps/details?id=com.etermax.apalabrados.lite&hl=es

https://itunes.apple.com/es/app/apalabrados/id441092257?mt=8

[6] *Rummykub app* Google Play and App Store

https://play.google.com/store/apps/details?id=com.rummikubfree&hl=es

https://itunes.apple.com/es/app/rummikub/id973113361?mt=8

[7] *RummyPlus* Google Play and App Store

https://play.google.com/store/apps/details?id=net.peakgames.mobile.rummi.android&hl=es

https://itunes.apple.com/es/app/rummy-plus/id690541689?mt=8

[8] *Qué es scrum?* Scrum.org

https://www.scrum.org/resources/blog/que-es-scrum

[9] Scrum of One

https://www.raywenderlich.com/162654/scrum-one-bring-scrum-one-person-operation

[10] Firebase

https://firebase.google.com/

[11] Ionic Framework

https://ionicframework.com/

[12] Angular.js

https://angularjs.org/

[13] *AngularJS Tutorial* W3schools

https://www.w3schools.com/angular/default.asp

[14] *Ionic issues* IonicTeam

https://github.com/ionic-team/ionic/issues

[15] *Old Sccramikub Report* Oriol Burgaya

https://upcommons.upc.edu/handle/2117/101200